

Programming: The Obstacle in Teaching Introductory Computer Science

Charles Dierbach

Department of Computer and Information Sciences
Towson University, Towson, Maryland 21252
cdierbach@towson.edu

The time has come in computer science education to put programming in its place. We, as educators, profess that the content of introductory courses convey concepts, and not just programming and other details. We retaliate if students refer to Introduction to Computer Science I as a “C++ course” or “Java course,” and then proceed to inform them that we are using a book whose title and table of contents appears to contradict that fact.

We must each ask ourselves, “What Concepts Am I Trying to Convey In The Introductory Courses?” My hope is that the answer is “primarily *general* concepts, and not just programming concepts and skills.” If not, why not? Is it the fact that your population of students are expected to become programmers, as opposed to theoreticians, and training programmers helps fulfill the dramatic shortage of IT workers? Is learning Java and web programming, in fact, what your students want?

The position taken here is that, consistent with the notion of higher education, we must be primarily concerned with training students to think - to think like a computer scientist, and generally educated person - from the very beginning courses. Therefore, *my* answer to ‘What Am I Trying to Convey’ is the conceptual framework of the field of computer science; how we see things, talk about things, and reason about things. This includes, but is not constrained by, programming issues and concepts.

If one accepts this position, the next question becomes “What Means Can Be Used to Convey A Conceptual Framework in an Educational Setting?” In general, I believe, there are two basic ways of doing this. One, by providing sufficient exposure to details, enabling students to formulate their own conceptual framework (i.e., “detail-driven” approach). The second, is to explicitly build a conceptual framework for students (i.e., a “concept-driven” approach) into which they can organize their detailed knowledge.

The current approach in introductory computer science courses is generally detail-driven, that learning to program is the means to the understanding of general concepts. On the contrary, the position taken here is that the concept-driven approach is the preferred approach, for the following reasons: (a) motivation and interest are more naturally generated from the understanding of a conceptual whole, (b) students with the most aptitude for the field will be properly motivated, and those with possibly insufficient aptitude (or just programming aptitude) will discover so early on in the curriculum, and (c) programming concepts that *are* covered will be understood with greater appreciation when presented in a larger conceptual context.

It is for these reasons that the role of programming in the introductory curriculum must be carefully *incorporated*, and not be made *encompassing* and therefore an obstacle to the task at hand.